2020 IEEE 6th Intl Conference on Big Data Security on Cloud (BigDataSecurity), IEEE Intl Conference on High Performance and Smart Computing, (HPSC) and IEEE Intl Conference on Intelligent Data and Security (IDS)

SDN-based Stateful Firewall for Cloud

Jian Li
School of Electronic Information.
Wuhan University
Wuhan, China
992807380@qq.com

Hao Jiang
School of Electronic Information.
Wuhan University
Wuhan, China
jh@whu.edu.cn

Wei Jiang
School of Electronic Information.
Wuhan University
Wuhan, China
1297690523@qq.com

Jing Wu
School of Electronic Information.
Wuhan University
Wuhan, China
wujing@whu.edu.cn

Wen Du
DS Information Technology Co., Ltd.
Shanghai, China
duwen@dscomm.com.cn

***Abstract*—The firewall service (FWaaS) is one of the key components of the cloud computing environment, it requires the ability to automatically open and flexibly adjust as needed. Current firewalls in the cloud are mainly based on static security rule configuration or simple rule matching, which makes them inflexible and cannot guarantee network security. In this paper, we propose a stateful firewall based on the programmable data plan, compared with existing SDN firewalls, our method is able to extract, analyze and record the connection state information of data packets in the data plane by designing a finite state machine and a state table. We implement a prototype using P4, then, evaluate its performance and overheads. The experimental results show that the scheme can achieve fine-grained access control and reduce communication overhead.**

Keywords*—*cloud computing, SDN, stateful firewall, P4, network security

I. INTRODUCTION

With the development and maturation of virtualization technology, the trend of cloud computing is unstoppable. In the Cloud, all kind of physical resources are virtualized. A tenant can rent the resources in the virtual resource pool according to their requirements.

Network security is a major factor in whether organizations choose to embrace or reject the cloud. On one hand, traditional attacks such as Distributed Denial of Service (DDoS), viruses and phishing still exist in clouds. On the other hand, new specific attacks on the computing mechanisms of cloud have also been found, including Economic Denial of Sustainability (EDoS) attack, cross Virtual Machine (VM) attack, and so on [1]. In order to prevent network services and end hosts from Internet attacks, a firewall, as one of the most critical security features, plays an important role in cloud computing environment. But the change of environment brings with it challenges that may not be fully addressed by the current generation of cloud firewalls. Traditional solutions are to place firewalls at the entry point of an Autonomous System(AS) [2], e.g., enterprise environment, known as static rule-based firewall, protects the internal nodes from attacks from the outside, the firewall as well as the internal nodes are managed by the same instance and considered safe. However, cloud computing blurs the concept of internal and external [3]. External parties can easily rent virtual instances residing in the perimeter and run their applications there. This calls for dynamic and fast firewall reconfiguration and replacement to adapt to VMs changes, which cannot be easily achieved by traditional firewalls. So a new firewall approach is needed in the cloud computing environment.

The integration of SDN (software defined Networking) [4] helps to solve the above issues. SDN is a new network design concept. By separating the control plane and forwarding plane of the network, and the programmable ability of the open network, it increases the flexibility, openness and innovation of the network, and has become the standard technology of cloud resource pool network construction. By introducing SDN, the data plane can act as a distributed stateless firewall. In order to further improve the security, the state firewall needs to be introduced. However, OpenFlow is stateless and limited in its ability to support customized protocols [5], the state management needs to be implemented in the controller. This causes the controller to be over-computing load and increases the communication overhead between the controller and the data plane, which in turn affects the efficiency of packet forwarding, and make it hard to guarantee QoS requirement specified by cloud firewall customers. With the rise of programmable data planes, network become greater flexibility, and P4 [6], as a data plane programming language, allows us to program the data plane, transforming the forwarding paradigm from a vendor-locked architecture to a user-definable architecture, regardless of the underlying protocols, packet headers, packet parsers, and packet processing behaviors can be defined in P4.

In view of the above problems and combined with the characteristics of programmable data plane, in this paper, we propose a state firewall method based on programmable data plane. In this method, by programming the data plane processing pipeline and modifying the matching table, the connection status can be recorded, tracked, and updated on the data plane, and global firewall functions can be managed on the control plane. Under the premise of ensuring the security of the SDN network, the network packet exchange between the control plane and the data plane is minimized.

The main contributions of this paper are summarized as follow: We firstly introduce the firewall architecture in SDN-based cloud to ensure the enterprise networks security. Second, we define a state table and finite state machine in the P4 data plane by programming and modifying the matching table of the data plane pipeline, in which the state detection and management is transferred from the controller to the data plane. Finally, we integrate a firewall provision module in the controller in order to manage the deployment of the Firewall applications. This integration enables interaction between administrators and data plane elements.

Project supported by the National Defense Basic Scientific Research Program of China (No. JCKY2018207C121); Shanghai Special Fund Project for Artificial Intelligence Innovation and Development under Grant 2018-RGZN-01013.

Authorized licensed use limited to: Mukesh Patel School of Technology & Engineering. Downloaded on July 11,2023 at 05:38:20 UTC from IEEE Xplore. Restrictions apply.

II. RELATED WORK

With the rapid development of cloud computing and the growth of user needs, the cloud service providers may be unreliable [7], Many data security and privacy events have been observed in cloud services today [8]. One reasonable solution is to protect the data locally on the end-user's device before sharing it through an untrusted cloud platform, there are many researches on data encryption algorithms [7-10]. While another is to protect it by deploying a security service such as a firewall, a ISP and so on, but traditional firewall have too many limitations and cannot meet cloud security needs [11]. In order to break through these limitations, SDN is an emerging model for centralizing network control and introduces the ability to program the networking behaviors. Tariq et al. [12] implemented a layer 2 firewall by modifying learning switch in POX controller according to MAC address, but the layer 3 and layer 4 traffic was not considered in their research. EnforSDN [13] proposes a management process that exploits SDN principles in order to separate the policy resolution layer from the policy enforcement layer in network service appliances. The concept improves the enforcement management, network utilization and communication latency, without compromising network policies. Besides, many SDN controllers propose their own version of stateless SDN firewall [14-15]. FleXam [16] is an extension of Openflow which integrates a stateless firewall. It runs on the controller and provides a means to specify a set of flow filters on specified parts of a packet. Then, it applies the associated action to the packets. However, they cannot handle stateful security applications like stateful firewalls.

Juan, W. [17] propose a SDN stateful firewall, the solution is based on Openflow and adds three new tables. This firewall keeps a table in the controller to save the connections' states and to synchronize the controller with the connection updates happened on the data plane tables. The other two tables are in the switch. One table manages the actual states of the connections and the other enables the data plane devices to process the next states. Zerkane S et al. [18] propose a SDN reactive stateful firewall. The application filters TCP communications according to the network security policies. It records and processes the different states of connections and interprets their possible transitions into OpenFlow (OF) rules. They use a reactive behavior in order to reduce the number of OpenFlow rules in the data plane devices and to mitigate some Denial of Service (DoS) attacks like SYN Flooding. The centralized SDN firewall is relatively easy to set up [19], but for delay-sensitive applications in cloud data, there is a possibility of overloading and the limit of scalability, which becomes a performance bottleneck.

In our scheme, we consider the factors of dynamic scalability, communicate cost, and efficiency to design an efficient stateful firewall system for the cloud environment.

III. SYSTEM ARCHITECTURE

A. Overview

As a firewall system for the cloud environment, the system should be scalable, flexible and efficient, meanwhile, in order to optimize the SDN centralized firewall method, provide fine-grained access control, reduce the communication overhead between the control plane and the data plane, and reduce the controller resource consumption. Here we introduce a novel SDN stateful firewall solution, which implements firewall as a service (FWaaS) in a cloud environment.

Figure 1 shows the overall system architecture. It has three parts: a cloud platform, a controller and a P4 data plane. The cloud platform provides the firewall service to the tenants, and collects the management and configuration requirements of the network and firewalls from the tenants, then delivers them to the controller through the neutron network component. The control plane is responsible for the management of the security policy between the network elements and the administrator. The main goal of the controller is to provide a global view of the network and dynamically deploy security policies on the data plane when needed. The Firewall service on the data plane is used to secure networks from unauthorized accesses, using the firewall policy configured by the controller to process packets. Each Instance of the Firewall is associated with a Controller and placed above its northbound interface.

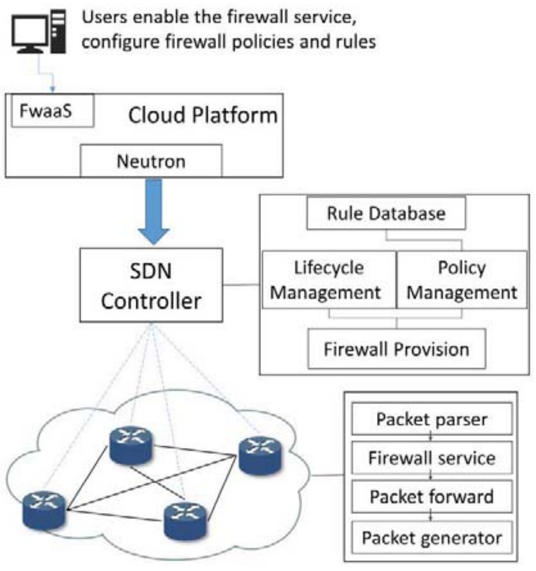

Fig. 1. System architecture

Firewall Provision: The firewall provision module is responsible for adding, modifying, and deleting firewall rules, and stores these rules in the rule database. It interacts with the controller, decides the firewall policy configured on each switch and whether the firewall function needs to be enabled, and then passes the configuration information to the controller for configuration, based on the data plane global operating state information obtained by the controller.

Packet parser: The packet parser module usually decodes the incoming data packets in the data plane. It parses the data according to the defined packet header structure. When a data packet enters the switch, the extract function is run to match the value of the packet header parameter and update it to the corresponding header instance to instantiate the packet header structure, and then pass the parsed data into the processing pipeline, or discard Eligible data.

Authorized licensed use limited to: Mukesh Patel School of Technology & Engineering. Downloaded on July 11,2023 at 05:38:20 UTC from IEEE Xplore. Restrictions apply.

Firewall Service: The firewall service module performs state detection of the data plane and updates of the state table. It contains a set of firewall rules. The rules specify the specific operations on the data packets. At the same time, the firewall service module needs to record the connection state of the network session between the two hosts, extract state information, then filter packets based on user-defined firewall rules and reserved connection state information, make a decision to allow forwarding or drop, and finally perform an update to the connection state table.

Packet Forward: The packet forward module determines the next hop information and port information according to the forwarding rules defined in the forwarding table. The forwarding table consists of matching fields, counters, and actions. The forwarding module matches the priority of flow entries in the forwarding table. For each packet entering the data plane, it checks its matching domain information. For packets that match successfully, it forwards them to the destination port and updates the counter. If the match fails, the data packet is sent to the controller as PACKET_IN message.

Packet Generator: The packet generator receives packets with metadata for packet generation. According to the previous decisions, the packet generator packages the packet with a modified header, and sends it out at the determined output port.

B. Data Plane Packet Processing

Stateful firewall inspects received packets and performs packet integrity checks by parsing and analyzing the structural information of the packet header. After the inspection is complete, the packet is dropped or sent to the next stage. The data header information is then checked to track active connection information in the session table, such as source IP address, destination IP address, application, port address, flag, serial number, and session state information. If a match is found in the state table, the firewall allows traffic to the network without applying a firewall policy and updating the session table before the packet is forwarded. If the current connection information is not found on the state table, the packet is filtered by a firewall rule. When a packet is allowed to enter the internal network, the corresponding session information is stored in the session table; if the firewall fails to filter, the packet will be dropped. The algorithm of stateful firewall filtering is shown in Algorithm 1.

Algorithm 1: Data Plane Stateful Firewall Algorithm

```
Initialize: session state table SST; session state check SSC; firewall
rule table FRT.
Input: received packet pkt.
Output: action of the packet.
1.   Pkt_Head = pkt.extract();
2.   Legitimacy = check_legitimacy(Pkt_Head);
3.   if Legitimacy == false then:
4.     Drop the packet;
5.   else:
6.     Session_Info = pkt.session();
7.   if NOT SST.find(Session_Info) then:
8.     if FRT.match(Session_info) == deny then:
9.       Drop the packet;
10.  else:
11.    Pkt_Transition = SSC.check(Session_Info);
12.    if Pkt_Transition == true then:
13.      SST.update(pkt);
14.      Apply_action(pkt);
15.    else:
16.      Drop the packet;
```

In our work we apply the algorithm to TCP communications. It has been instantiated with TCP states, transitions and actions. When a session is initiated from the internal network, the stateful firewall allows data packets from the external network to be sent to the internal network. This can be determined by checking the flag of the three-way handshake process of TCP to decide whether to allow forwarding. The host in the internal network sends a SYN-flag TCP packet to the external host requesting the connection. The host in the external network responds with SYN-ACK-flag TCP, and the internal host responds with ACK-flag TCP. At this point, the TCP three-way handshake is completed and establish connection. When the session is terminated, the host sends a FIN-flag TCP packet, and the state firewall checks the FIN flag to stop allowing packets from the external host to be sent to the internal host.

C. State Management Mechanism

In order to detect and manage the connection state, we need to consider the following issues. The first is how to maintain the state table locally in the data plane and reduce the consumption of local storage resources; the second is how to check the legitimacy of the current packet when the packets of different phases of the session arrive; the last is how to initiate, transfer and update the state table.

State Table Creation: First of all, in order to implement the state firewall function, we need to record the communication process and update the connection state information. To do this, we need to design a state table in the data plane. The state table consists of matching fields, connection state, and timeout time. It maintains a state for each data flow and sets the timeout period. If a certain state is not queried, a destruction operation is performed to release the memory space.

In response to the above TCP communication problem, the matching field is composed of a quad (IP_src, IP_dst, TCP_src, TCP_dst). At the same time, in order to save data plane memory, we have hashed the matching field.

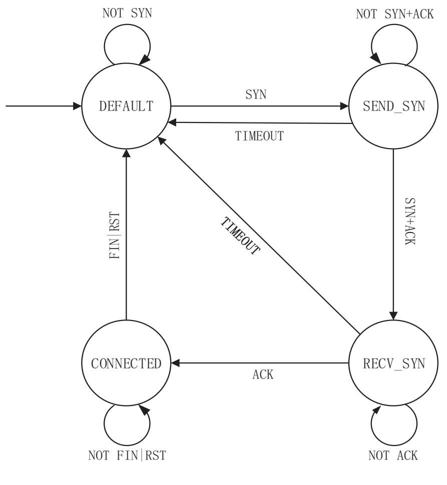

Fig. 2. Connection state transition

State Transition: Secondly, in order to enable the data plane device to detect whether the data packets at different stages of the network session between the two hosts are legal and to update and manage the state table at the same time, we

Authorized licensed use limited to: Mukesh Patel School of Technology & Engineering. Downloaded on July 11,2023 at 05:38:20 UTC from IEEE Xplore. Restrictions apply.

have modified the pipeline logic to add a finite state machine for state detection.

Figure 2 shows a finite state machine for identifying a three-way handshake. It is an abstract model containing an initial state (e.g., default) and a quad (S, I, O, T), where S is a finite set of state, I is a finite set of input events, O is a finite set of output actions, and T：$S \times I \rightarrow S \times O$ is a conversion function that maps states and events to states and action pairs. It matches the header information of the data packet with the corresponding state in the state table. When the match is successful, it returns the next stage state information and determines that the data packet is a legitimate connection. Otherwise, it determines that the data packet is illegal and performs a discard operation.

State Table Update: The last is how the state table is updated and managed as the session progresses. The first is when the data packet arrives, the state table is queried according to the matching field. If there is no record, the state is set to DEFAULT. After obtaining the state of the data packet, the state information and the flag bit information in the data packet are passed to the finite state machine for detection. If the state transition of the state machine is satisfied, the next connection state information is returned, and the state table update operation is performed based on the state information, including creating, changing, and deleting entries.

IV. EVALUATION

In this section, we test the system from two aspects. Firstly, we deploy a simple system to verify the system feasibility. In the experiment, we deploy the virtual network on the Openstack platform and Mininet. We will use the stateful firewall to control the internal network to access the external network. Secondly, we test the system performance through simulation experiments.

A. Experiment Setup

In this experiment, we check whether the system can operate normally and implement a state-based fine-grained stateful firewall. During the connection establishment process, the switch makes decisions based on the state information of the data packets and the information recorded in the state table to determine the connection. Whether it complies with the firewall rules and performs state transfer according to different state conditions to update the record of the state table. We have established a simple topology with multiple hosts and 4 switches, as shown in Figure 3. Among them, the host h1 is on the internal network, and the other hosts are on the external network.

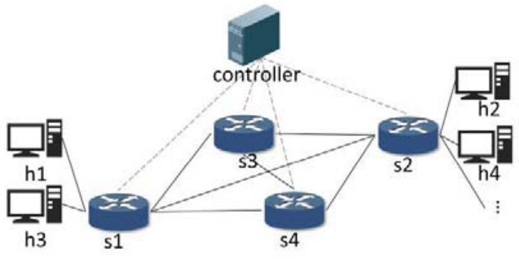

Fig. 3. Test topology

In the test, two operations of h1 initiating a TCP connection to h2 and h2 initiating a TCP connection to h1 were respectively implemented. Only the internal h1 can establish a TCP connection with the external h2 and communicate. After the connection is completed, h2 cannot connect to h1. This verifies that the system can provide fine-grained state-based access control to ensure the security of internal hosts.

B. Performance Evaluation

In this experiment, we evaluate the update time of the state firewall rule and the communication overhead of the state firewall during the communication process with the control plane, because the rule is updated every time when the internal host initiates a connection request or completes the connection, it takes some time to update the rule, which will affect the TCP connection time. We compare our approach with stateless firewall and stateful firewall on controller to evaluate its performance. The rules of stateless firewalls are to allow communication between specific addresses of internal and external networks. We use a switch and an external host as the Web server to set up the experiment. In the two cases, the clients generate the same number of simultaneous TCP connections. We started the tests with 10 simultaneous connections and ended at 100 simultaneous connections per second in a continuous and a constant interval of time.

We analyze the data with two objectives. The first one is the performance of the firewall compared with a controller without a firewall and a controller firewall. In this case, we focus on the scalability of the firewall by observing the evolution of the packet processing time zones with the number of simultaneous connections. We find the total processing time and the results are shown in Figure 4. In the second case, we are interested in observing how much extra time the Firewall needs to process the packet-in and the connections. Performance results are presented in Figure 5.

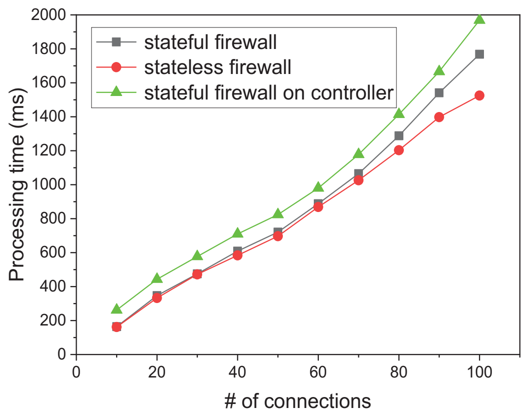

Fig. 4. Processing time according to the number of connections

Figure 4 shows the total connection time as a function of the number of internal hosts. The results show that the connection time of our method is slightly longer than the connection time in stateless firewall, but it is significantly reduced compared to the stateful firewall on controller. The additional delay is because the SDN switch must update the state information to allow packets from external hosts. And

Authorized licensed use limited to: Mukesh Patel School of Technology & Engineering. Downloaded on July 11,2023 at 05:38:20 UTC from IEEE Xplore. Restrictions apply.

once the connection is terminated, it must delete the state information to reject packets from external hosts.

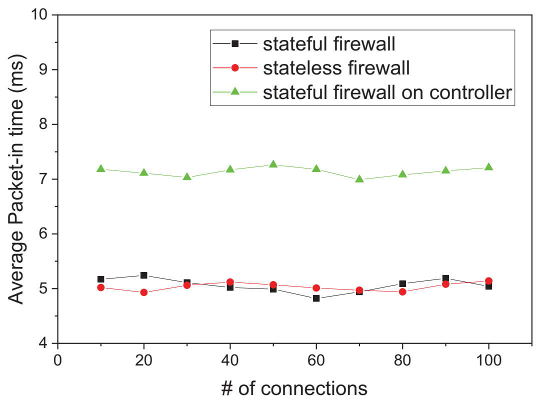

Fig. 5. Average packet-in time according to the number of connections

Figure 5 shows the change of the average packet-in time with the number of internal hosts. We observe in both experiments a rather constant average time. The average packet-in time of the stateful firewall and the stateless firewall stays between 4.5 ms and 5.6 ms, furthermore for the firewall on controller it is almost constant around 0.5 ms. The results show that our method has little effect on the average Packet-in time, this is because the state firewall makes corresponding decisions based on the state information retained by the data plane, thereby avoiding additional communication with the control plane.

V. CONCLUSION

Cloud computing security is of great importance today. In this paper, we proposed a distributed stateful firewall scheme to increase the security of the cloud environment. To avoid controller overhead and reduce the communication overhead between the control plane and the data plane, we implement state recording, detection, and integration by adding a firewall service module in the data plane, meanwhile, use the controller to manager firewall lifecycle and policy. Finally, we implemented the system in a simulation environment. After simulations, the results show that the scheme can achieve fine-grained state detection. At the same time, at the beginning and end of the session, the controller does not have to introduce additional loads to change the rules in the switch. This process only needs to be done on the data plane and adds little overhead.

REFERENCES

[1] Singh S , Jeong Y S , Park J H . A Survey on Cloud Computing Security: Issues, Threats, and Solutions[J]. Journal of Network and Computer Applications, 2016, 75:200-222.J.

[2] Ahmed, Patel, Mona, et al. An intrusion detection and prevention system in cloud computing: A systematic review[J]. Journal of Network and Computer Applications, 2013.

[3] J. Cropper, J. Ullrich, P. Frühwirt and E. Weippl, "The Role and Security of Firewalls in IaaS Cloud Computing," 2015 10th International Conference on Availability, Reliability and Security, Toulouse, 2015, pp. 70-79.

[4] Software-Defined Networking: A Comprehensive Survey[J]. Proceedings of the IEEE, 2015, 103(1):14-76.

[5] Jararweh Y , Alayyoub M , Ala' Darabseh, et al. Software defined cloud: Survey, system and evaluation[J]. Future Generation Computer Systems, 2016, 58(3):56-74.

[6] Bosshart P , Daly D , Izzard M , et al. Programming Protocol-Independent Packet Processors[J]. ACM SIGCOMM Computer Communication Review, 2013, 44(3):87-95.

[7] H. Qiu, M. Qiu, G. Memmi and M. Liu, "Secure Health Data Sharing for Medical Cyber-Physical Systems for the Healthcare 4.0," in IEEE Journal of Biomedical and Health Informatics.

[8] H. Qiu, H. Noura, M. Qiu, Z. Ming and G. Memmi, "A User-Centric Data Protection Method for Cloud Storage Based on Invertible DWT," in IEEE Transactions on Cloud Computing.

[9] H. Qiu, M. Qiu, M. Liu and Z. Ming, "Lightweight Selective Encryption for Social Data Protection Based on EBCOT Coding," in IEEE Transactions on Computational Social Systems, vol. 7, no. 1, pp. 205-214, Feb. 2020.

[10] G. O. Karame, C. Soriente, K. Lichota, S. Capkun, "Securing cloud data under key exposure", IEEE Trans. Cloud Comput., vol. 7, no. 3, pp. 838-849, Jul./Sep. 2017.

[11] Bays L , Oliveira R , Barcellos M , et al. Virtual network security: threats, countermeasures, and challenges[J]. Journal of Internet Services and Applications, 2015, 6(1):1.

[12] Javid T , Riaz T , Rasheed A . A layer2 firewall for software defined network[C]// 2014 Conference on Information Assurance and Cyber Security (CIACS). IEEE, 2014.

[13] Y. Ben-Itzhak, K. Barabash, R. Cohen, A. Levin and E. Raichstein, "EnforSDN: Network policies enforcement with SDN," 2015 IFIP/IEEE International Symposium on Integrated Network Management (IM), Ottawa, ON, 2015, pp. 80-88.

[14] Suh M , Park S H , Lee B , et al. Building firewall over the software-defined network controller[C]// 2014 16th International Conference on Advanced Communication Technology (ICACT). IEEE, 2014.

[15] F. Canellas, A. Mimidis, N. Bonjorn and J. Soler, "Policy Framework Prototype for ONOS," 2019 IEEE Conference on Network Softwarization (NetSoft), Paris, France, 2019, pp. 218-222.

[16] Shirali-Shahreza, S., Ganjali, Y.: FleXam: Flexible sampling extension for monitoring and security applications in OpenFlow. In: Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, pp. 167–168 (2013)

[17] Juan, W., Jiang, W., Shiya, C., Hongyang, J., Qianglong, K.: SDN (self-defending network) firewall state detecting method and system based on OpenFlow protocol. China Patent CN104104561 A, 11 August 2014

[18] Salaheddine Zerkane, David Espes, Philippe Le Parc, 等. Software Defined Networking Reactive Stateful Firewall[C]// IFIP International Information Security and Privacy Conference. Springer International Publishing, 2016.

[19] Q. Yan, F. R. Yu, Q. Gong and J. Li, "Software-Defined Networking (SDN) and Distributed Denial of Service (DDoS) Attacks in Cloud Computing Environments: A Survey, Some Research Issues, and Challenges," in IEEE Communications Surveys & Tutorials, vol. 18, no. 1, pp. 602-622, Firstquarter 2016.

Authorized licensed use limited to: Mukesh Patel School of Technology & Engineering. Downloaded on July 11,2023 at 05:38:20 UTC from IEEE Xplore. Restrictions apply.